Ayrık Zamanlı Kayan Kipli Denetimin Kayan Nokta Aritmetiği ile FPGA Üzerinde Gerçeklenmesi

Implementation of Discrete Time Sliding Mode Control with Floating Point Arithmetic on an FPGA

Mehmet Muzaffer KÖSTEN[1], *Mehmet Önder EFE*[2]

[1]Bilgisayar Mühendisliği Bölümü
Niğde Üniversitesi, Niğde
mmkosten@ieee.org

[2]Bilgisayar Mühendisliği Bölümü
Hacettepe Üniversitesi, Ankara
onderefe@hacettepe.edu.tr

Özetçe

Bu çalışmada kayan kipli kontrol tekniği FPGA üzerinde kayan noktalı sayılar kullanılarak gerçekleştirilmiştir. Tasarlanan sistem ilk olarak IEEE754 standardına uygun olarak 32 bit olarak gerçeklenmiş ve elde edilen sonuçlar Matlab ortamında yapılan hesaplamalarla karşılaştırılmıştır. FPGA üzerinde gerçekleştirilen sistem azaltılmış bit uzunluğuna sahip kayan noktalı sayılar ile tekrar tasarlanmış ve sonuçlar incelenmiştir. Elde edilen sonuçlar donanım kaynak tüketimi ve sonuç doğruluğu açısından karşılaştırılmıştır.

Abstract

In this study, sliding mode control techniques is implemented on FPGA using floating point numbers. The designed 32 bit system is implemented in compliance with IEEE754 standard and the results are compared with those obtained using Matlab simulations. The system on FPGA is reimplemented with reduced number of floating point representation and the results are analyzed. A comparison in terms of the resource usage and the precision is presented.

1. Giriş

Kontrol sistemlerinin donanım üzerinde gerçekleştirilmesi gün geçtikçe daha çok önem kazanmaktadır. Bunun temel sebebi donanım kabiliyetleri artarken maliyetlerin düşmesidir. Otomatik kontrol sistemlerinin donanım üzerinde gerçekleştirilmesi ise çeşitli sebeplerle ilginç bir konudur. Bunların başında tüm sistemin ayrık zamanlı olması, dolayısıyla analiz ve tasarımın ayrık zamanlı sistemler kuramının kurallarına göre yapılmasının gerekli olmasıdır. Bir diğer sebep ise düşük maliyetli sistemlerde doğrusal olmayan terimlerin gerçekleştirilmesi için çeşitli nümerik tekniklere ihtiyacın duyulması, bunun için özel olarak çalışılmasının gerekliliğidir. Bir diğer sebep ise sayıların gösterimi ile birlikte gelen tasarım seçeneklerinin incelenmesine duyulan ihtiyaçtır. Bu bildiri, kayan kipli kontrol yöntemini FPGA üzerinde koşturarak kayan kipli aritmetik ile hassasiyetin nasıl etkilendiğini donanım üzerinde yapılan deneylerle ele almaktadır. Kayan kipli kontrol tekniği geçmişte pek çok uygulamada başarıyla kullanılmıştır. Bu yaygın kullanımına karşılık FPGA tabanlı kayan kipli kontrol uygulamaları ile ilgili literatürde çok fazla çalışma bulunmamaktadır. Mevcut çalışmalar ise çoğunlukla motor kontrolü, robot kol kontrolü, evirici ve çeviri tasarımlarının kontrolü gibi amaçlarla gerçekleştirilmiştir.

Hace ve Franc kayan kipli kontrol algoritmasını teleoperasyon kontrolü için FPGA üzerinde gerçekleştirmiştir [1]. Bir diğer çalışmada Lentijo vd. yüksek frekanslı güç çevirici kontrolünde FPGA tabanlı kayan kipli kontrolcü kullanmıştır, [2]. Benzer bir çalışma Li vd. tarafından yüksek frekanslı evirici kontrolü için FPGA üzerinde yapılmıştır, [3]. Motor kontrolü ile ilgili birkaç çalışma da Lin vd. tarafından gerçekleştirilmiştir, [4]–[6]. Kayan kipli gözleyici tasarımının FPGA üzerinde gerçekleştirilmesi ile ilgili bir çalışma Lienhardt vd. tarafından sunulmuştur, [7]. Başka bir çalışmada FPGA tabanlı kayan kipli denetleyici tasarımı ile ilgili bir yöntem Piltan vd. tarafından önerilmiştir, [8].

Anılan çalışmaların büyük çoğunluğunda sabit noktalı[1] sayı biçimi kullanılmış olup birkaç çalışmada ise sayılar genişletilerek (örneğin 1000000 ile çarparak) tam sayı biçimine sokulmuş ve hesaplamalar bu şekilde gerçekleştirilmiştir. Bu tarz yaklaşımların kullanılmasının temel nedeni FPGA donanım kaynaklarının dikkatli bir şekilde kullanılma ihtiyacıdır. Bu yaklaşım önceki nesil FPGA aileleri için mantıklı olmakla birlikte işlem hassasiyeti konusunda birtakım sorunlar ortaya çıkarmaktadır. İşlem hassasiyetini korumak için kayan noktalı sayılarla işlem yapmak iyi bir tercih olmakla birlikte kayan noktalı sayı aritmetiğinin donanım kaynak tüketiminin diğer sayı biçimlerine göre daha fazla olması hassasiyet ve kaynak tüketimi açısından bir dengeli bir tasarım yönteminin izlenmesini zorunlu kılmaktadır. Bu bildiride sunulan çalışmalar da bu soruya bir cevap bulma gayesi gütmektedir.

[1] İng. *fixed point*

Gelişen teknoloji ile birlikte FPGA'ların sahip olduğu donanım kaynak miktarı artmış ve ciddi bir sorun olmaktan çıkmıştır. Ayrıca yeni nesil FPGA aileleri sahip oldukları donanımsal hızlandırıcılar ile kayan noktalı sayı işlemlerini az kaynak tüketerek ve hızlı bir şekilde gerçekleştirebilmektedir.

Bu çalışma kapsamında FPGA üzerinde uyarlanan sistem daha önce yapılan çalışmalardan farklı olarak tamamen kayan noktalı sayılar kullanılarak tasarlanmıştır. Sistem ilk olarak Matlab üzerinde gerçekleştirilmiş ve elde edilen sonuçlar FPGA üzerinde çalışan sistem ile karşılaştırılmıştır. Karşılaştırmalar yapılırken farklı bit uzunluklarına sahip kayan noktalı sayılar kullanılmıştır.

Bu bildiri şu şekilde düzenlenmiştir: İkinci bölümde kayan noktalı sayılar, üçüncü bölümde sistem modeli ve kontrol sisteminin tasarımı anlatılmış, elde edilen sayısal sonuçlar tartışılmıştır. Sonuç ifadeleri ise dördüncü bölümde verilmiştir.

2. Kayan Noktalı Sayılar

Kayan noktalı sayı biçimi gerçek sayıların bilgisayar ortamında gösterim şeklidir. Kayan noktalı sayı gösterimi sabit noktalı sayılara göre daha geniş sayı aralığında işlem yapmakta fakat buna karşılık ihtiyaç duyduğu donanım kaynağı miktarı yüksek olmaktadır. Bu sayı formatı IEEE-754 standardı olarak da kabul edilmektedir. Bu standartlar çerçevesinde iki kullanım biçimi mevcuttur:

- Tek Duyarlı Gösterim[1] (32-Bit)
- Çift Duyarlı Gösterim[2] (64-Bit)

Kayan noktalı sayıların ifade biçimi (1) ve (2) ile verilmiştir.

$$\text{Sayı} = (-1)^{s}(1.f)\cdot 2^{e-bias} \tag{1}$$

$$e - bias = floor(\log_2(\text{Sayı})) \tag{2}$$

- "*s*" ifadesi işaret bitini göstermektedir. İfadenin değeri 0 ise sayı pozitif, 1 ise sayı negatiftir.
- "*bias*" ifadesi IEEE-754 standardında tanımlanmış olup, üs sayısından çıkarılan değeri ifade etmektedir. 32 bit gösterim için bu değer 127 olarak belirlenmiştir.
- "*f*" ifadesi ise virgülden sonra gelen kesir kısmı ifade etmektedir.

IEEE-754 standardına göre kayan noktalı sayı gösteriminde en anlamlı bit işaret biti olarak atanmış, ondan sonra gelen sekiz bit üssel kısım için atanmış ve kalan 23 bit ise kesir kısmının gösterimi için ayrılmıştır. Belirtilen özellikler 32 bit uzunluğundaki IEEE-754 standardına ait olup, istendiği takdirde bit uzunlukları değiştirilerek değişik uzunlukta kayan noktalı sayılar Denklem (1) ve (2) kullanılarak elde edilmektedir. Değişik bit uzunluklarına sahip kayan noktalı sayıların FPGA üzerindeki başarımına dair bir çalışma Lee vd. tarafından gerçekleştirilmiştir, [9].

3. Sistem Modeli

Gerçekleştirilen sisteme ait matematiksel ifadeler sırasıyla Denklem (3)-(15) ile verilmiştir.

[1] İng. *single precision*
[2] İng. *double precision*

Ayrık zamanlı sistemin dinamiği (3)-(4) ile verilmektedir. Bu gösterim, T_s örnekleme periyodunu göstermek üzere sürekli zamanlı bir sistem dinamiğinden türetilmiştir. Sistemin durumları X_1 ve X_2 değişkenleri durum değişkenlerini, k alt indisi ise zamanın ayrık değerlerini göstermektedir.

$$X_{1(k+1)} = X_{1(k)} + Ts * X_{2(k)} \tag{3}$$

$$X_{2(k+1)} = X_{2(k)} + Ts(f_k + \Delta f + g_k u_k + \Delta g u_k) \tag{4}$$

Dinamik gösterimi yukarıda verilen sistemin tasarıma esas oluşturacak doğrusal olmayan terimleri f_k ve g_k fonksiyonlarının içinde, bu fonksiyonlarla ilgili kesinsizlikler ise Δf ve Δg fonksiyonlarının içinde ifade edilecektir. Bu fonksiyonlara ilişkin analitik ayrıntılar aşağıda verilmektedir.

$$f_k = f(x_{1(k)}, x_{2(k)}) = x_{1(k)}\cos(x_{2(k)}) - {x_{2(k)}}^2 \tag{5}$$

$$\Delta f = 0.1\{sin^3(x_{2(k)}) + \cos(x_{1(k)})\} \tag{6}$$

$$g_k = g(x_{1(k)}, x_{2(k)}) = \frac{1 + {x_{2(k)}}^2 \sin^2(x_{1(k)})}{1 + {x_{1(k)}}^2 + {x_{2(k)}}^2} \tag{7}$$

$$\Delta g = \frac{0.01}{1 + (x_{2(k)}\cos x_{1(k)})^2} \tag{8}$$

Bu bilgiler ışığında, aşağıdaki referans sinyalleri seçilmiş,

$$r_{1(k)} = \sin\left(\frac{2\pi k T_s}{T}\right) \tag{9}$$

$$r_{2(k)} = \frac{2\pi}{T}\cos\left(\frac{2\pi k T_s}{T}\right) \tag{10}$$

bu referans sinyalleri ve durum değişkenleri arasında aşağıdaki hata terimleri tanımlanmış

$$e_{1(k)} = x_{1(k)} - r_{1(k)} \tag{11}$$

$$e_{2(k)} = x_{2(k)} - r_{2(k)} \tag{12}$$

ve erişme kuralı aşağıdaki gibi seçilmiştir.

$$s_{(k+1)} = (1 - \lambda_1 T_s)s_{(k)} - \lambda_2 T_s \text{sgn}(s_{(k)}) \tag{14}$$

Bu seçimler (15) ile verilen kontrol sinyalinin türetilmesini sağlayacaktır. Bu sinyal nominal sistem modelinde mevcut olan bilgiler kullanılarak oluşturulmuş, (3)-(4) ile verilen tam sistem üzerinde test edilerek farklı nümerik gösterimlerin etkileri incelenmiştir.

$$u_k = \frac{1}{T_s g_k}((1 - \lambda_1 T_s)s_{(k)} - \lambda_2 T_s \text{sgn}(s_{(k)}) + r_{2(k+1)} - \lambda_3 e_{1(k+1)} - X_{2(k)} - T_s f_k) \tag{15}$$

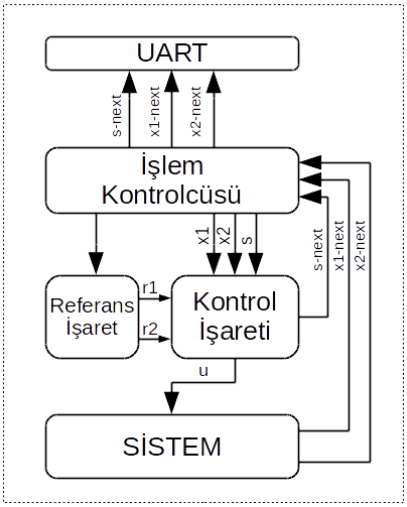
Yukarıdaki ifadede işaret fonksiyonunun getireceği olumsuzlukları bertaraf etmek için aşağıda verilen yaklaştırma da donanım tanımlama dili ile gerçeklenmiş ve kod içerisinde kullanılmıştır.

$$\text{sgn}(s_{(k)}) \cong \frac{s_{(k)}}{0.01 + |s_{(k)}|} \tag{16}$$

Verilen eşitliklerde $\lambda_1 = \lambda_2 = \lambda_3 = 1$ olarak seçilmiş olup $T_s = 0.01$ ve $T = 4$ olarak kullanılmıştır.

3.1. Ölçümler

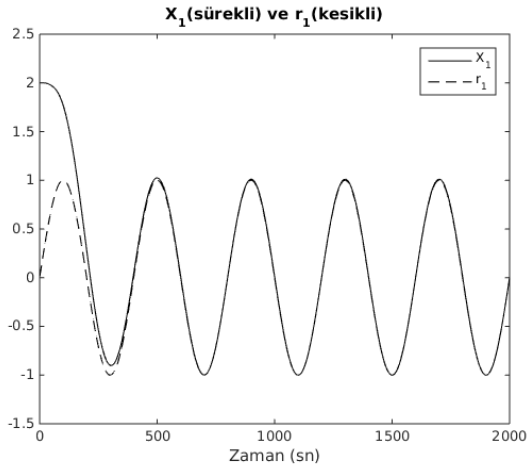
Verilen eşitlikler kullanılarak Matlab ve FPGA üzerinde sistem gerçekleştirilmiştir. FPGA üzerinde gerçekleştirilen sistem için "Vivado System Generator for DSP" yazılımı kullanılarak Xilinx Artix 7 FPGA'sı için VHDL dili ile tasarım yapılmış olup tasarlanan sistem NEXYS4 deneme kartı üzerinde çalıştırılmıştır. Gerçekleştirilen sistemde bulunan bileşenlere ait genel gösterim Şekil 1'de verilmiştir.



Şekil 1: FPGA tasarımının genel görünümü.

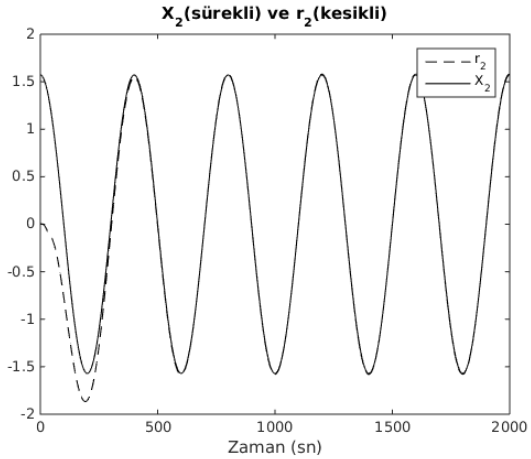
Sistem için gerekli olan referans işaretleri FPGA üzerinde LUT[1] kullanılarak üretilmiş ve sisteme girilmiştir. Bu sayede Matlab ile üretilen referans işaretleri ile FPGA üzerindeki sisteme girilen referans işaretlerinin birebir aynı olması sağlanmıştır. Her adımda hesaplanan değerler UART arabirimi ile bilgisayara gönderilmiştir.

IEEE-754 standardına uygun olan 32 bit kayan noktalı sayılar kullanılarak yapılan tasarımda Matlab ve FPGA üzerinde çalıştırılan sistem neredeyse aynı sonuçları üretmiştir. Elde edilen sonuçlar Şekil 2-4 ile verilmiştir.

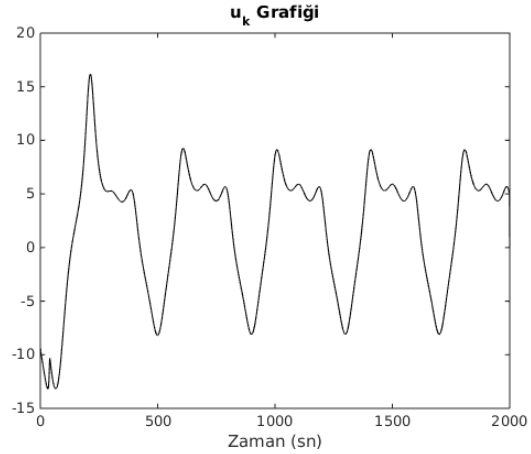


Şekil 2: Matlab ve FPGA için X_1 ve r_1 grafiği

[1] İng. Kıs. *Look-Up Table*

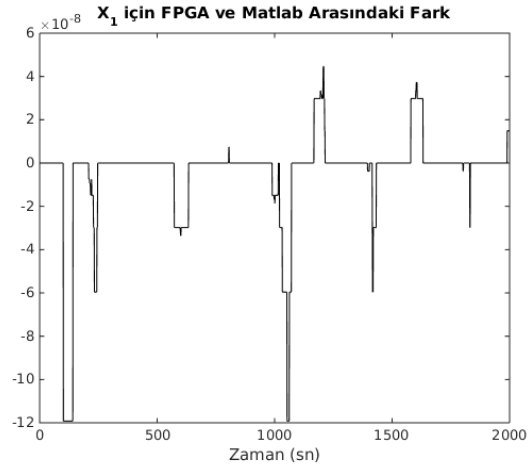


Şekil 3: Matlab ve FPGA için X_2 ve r_2 grafiği

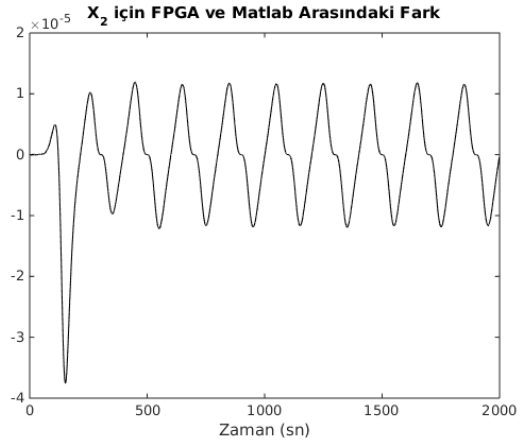


Şekil 4: Matlab ve FPGA için $u_{(k)}$ grafiği.

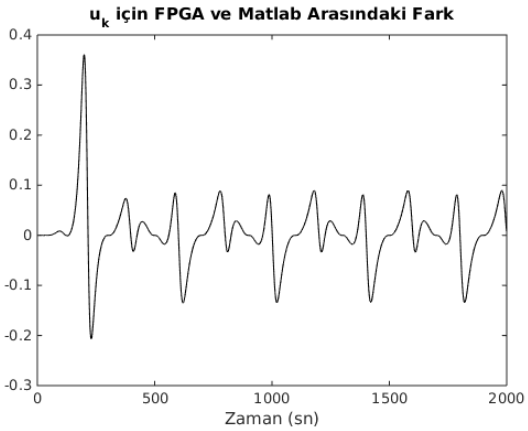
Sonuçlar grafik üzerinde örtüşmekle beraber, FPGA ve Matlab sonuçları arasında grafik üzerinde görülemeyecek kadar küçük farklılıklar mevcuttur. Bu farklara ait çizimler sırasıyla Şekil 5, 6 ve 7'de verilmiştir.



Şekil 5: X_1 için FPGA ve Matlab üzerinde hesaplanan değerler arasındaki fark.



Şekil 6: X_2 için FPGA ve Matlab üzerinde hesaplanan değerler arasındaki fark.



Şekil 7: $u_{(k)}$ için FPGA ve Matlab üzerinde hesaplanan değerler arasındaki fark.

Şekil 5, 6 ve 7 ile verilen grafiklerden görüleceği üzere FPGA ve Matlab sonuçları arasındaki fark X_1 için 10^{-8} ve X_2 için 10^{-5}mertebelerindedir. $u_{(k)}$ içinse ilk başta bulunan tepe değerleri dışında yaklaşık olarak ±0.1 aralığında değişmekte olup en fazla farkın olduğu durumda bile fark 0.4'e ulaşmamaktadır.

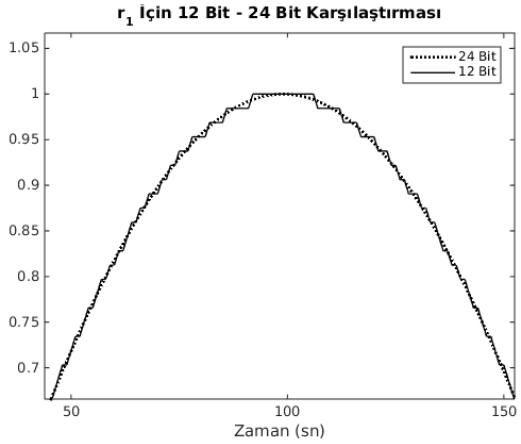
3.2. Azaltılmış Bit Kayan Noktalı Sayı Ölçümleri

Aynı ölçümler, Lee ve diğerlerinin yaptığı çalışmada belirttikleri 24, 20, 16 ve 12 bit uzunluğundaki kayan noktalı sayılar kullanılarak tekrarlanmıştır, [9]. İlgili bit uzunluklarına ait bilgiler Tablo 1'de verilmiştir.

Tablo 1: 24, 20, 16 ve 12 Bit Kayan Noktalı Sayılar için bit uzunlukları [9].

Biçim	Bit Uzunlukları		
	İşaret Kısmı	Üs Kısmı	Kesir Kısmı
24Bit	1 Bit	6 Bit	17 Bit
20Bit	1 Bit	6 Bit	13 Bit
16Bit	1 Bit	6 Bit	9 Bit
12Bit	1 Bit	6 Bit	5 Bit

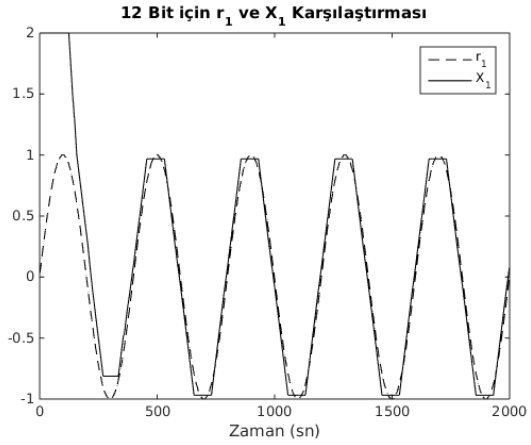
Kesirli kısımda yapılan bit azaltmasının oluşturduğu etki r_1 işaretinin 24 bit ve 12 bit olarak hesaplanması ile Şekil 5'te verilmiştir. Etkinin daha iyi gözlenebilmesi için r_1 işaretinin bir kısmı ayrıntılı olarak çizdirilmiştir.



Şekil 8: r_1 işareti için 12 bit – 24 bit karşılaştırması

Bit uzunluğunun azaltılması ile oluşan etki, sistemde yapılan hesaplamalar ile birikerek çıkışta değişime neden olmaktadır. 16 bit uzunluğuna kadar olan hesaplamalarda oluşan etki çıkışta çok fazla değişim oluşturmamakta olup, kesir kısmının en az olduğu 12 bit uzunluğu için gözle görülür bir fark oluşturmaktadır.

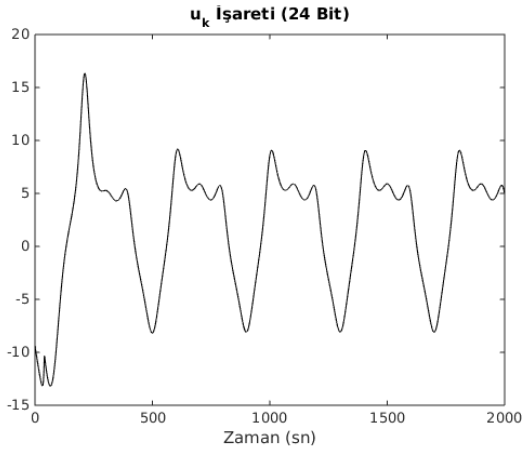
Bu etkiye dair çizim r_1 referans işareti ve buna bağlı olarak hesaplanan X_1 işaretinin 12 bit olarak hesaplanması ile Şekil 9'da verilmiştir.



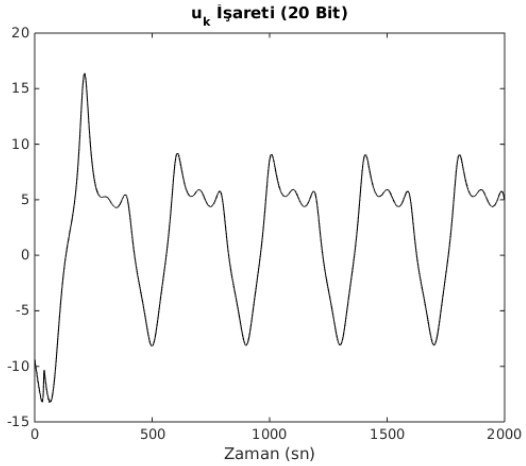
Şekil 9: 12 bit için r_1 ve X_1 grafiği.

Şekil 9'dan görüleceği üzere özellikle değişimin fazla olduğu tepe noktalarında 12 bit gösterim yetersiz kalmakta ve kesime gitmektedir. 12 bit gösterimde kesir kısmı için kullanılan 5 bit bu hesaplamalar için yetersiz kalmaktadır.

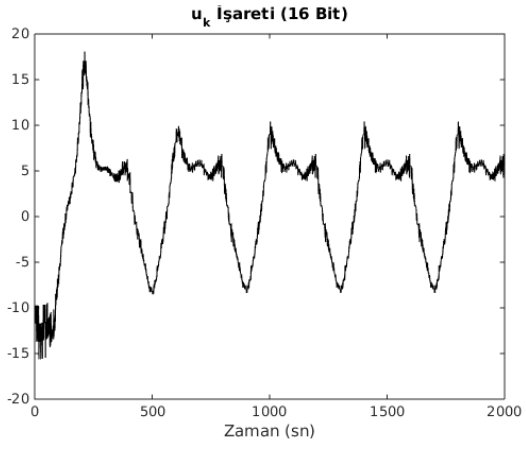
Farklı bit uzunlukları için hesaplanan $u_{(k)}$ kontrol işaretine ait grafikler sırasıyla Şekil 10-13 ile verilmiştir.



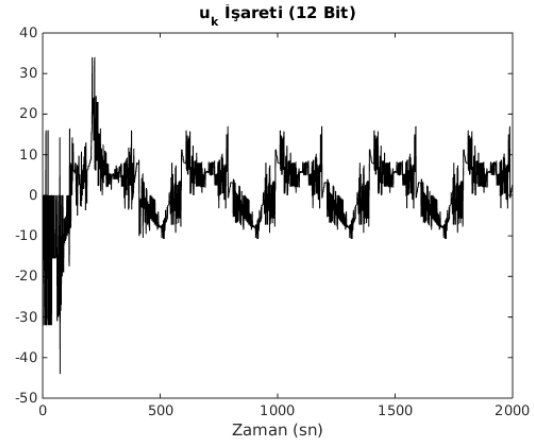
Şekil 10: 24 bit gösterim için $u_{(k)}$ işaretinin davranışı.



Şekil 11: 20 bit gösterim için $u_{(k)}$ işaretinin davranışı.



Şekil 12: 16 bit gösterim için $u_{(k)}$ işaretinin davranışı.



Şekil 13: 12 bit gösterim için $u_{(k)}$ işaretinin davranışı.

Şekil 10 ve 11 ile verilen grafiklerden görüldüğü üzere 20 ve 24 Bit gösterim için $u_{(k)}$ işaretinde gözle görülür bir bozulma olmamakta olup sonuçlar Şekil 4 ile verilen grafiğe benzemektedir.

Şekil 12 ile verilen grafikte ise Şekil 4'e göre birtakım bozulmaların olduğu görülmektedir. Bununla beraber bozulmaların işaretin genel karakteristiğini çok değiştirmediği görülmektedir.

Verilen bit uzunlukları içerisinde en fazla bozulma Şekil 13'ten görüleceği üzere 12 bit uzunluğundaki gösterimde gözlemlenmektedir. Özellikle tepe noktalarında oluşan bozulmalar Şekil 4 ile verilen işaret davranışından oldukça farklılık göstermektedir.

3.3. FPGA Kaynak Tüketimi

FPGA için tasarlanan sistem Xilinx Artix-7 ailesinden XC7A100T FPGA modeli için 100 MHz'de çalışacak şekilde sentezlenmiştir. FPGA üzerinde çalışan sistem bir iterasyon için 57 saat çevrimine ihtiyaç duymakta olup 100 MHz'de çalışan sistem için bu 0.57 µs (570 ns) olan bir işlem süresine denk gelmektedir. FPGA üzerinde çalışan sisteme ait kaynak tüketim bilgileri Tablo 2 ile verilmiştir.

Tablo 2: Farklı bit uzunlukları için donanım kaynak tüketimleri

Donanım Kaynakları	**Bit Uzunluğu - Kaynak Tüketimi (%)**				
	32Bit	**24Bit**	**20Bit**	**16Bit**	**12Bit**
FF[1]	10,38	7,10	5,84	4,37	3,41
LUT[2]	33,06	24,33	18,01	12,84	8,63
Bellek LUT	1,21	1,27	0,86	0,56	0,39
I/O	1,43	1,43	1,43	1,43	1,43
BRAM[3]	2,96	2,96	2,96	2,96	2,96
DSP48	32,08	17,08	11,67	11,67	11,67
BUFG[4]	3,13	3,12	3,12	3,12	3,12

[1] İng. Kıs. *flip-flop*
[2] İng. Kıs. *look-up table*
[3] İng. Kıs. *block random access memory*
[4] İng. Kıs. *global buffers*

4. Sonuçlar

Verilen grafikler incelendiğinde azaltılmış bit uzunluğuna sahip kayan noktalı sayılar ile tasarlanmış kontrol sisteminin, IEEE-754 standardına göre (32 bit) tasarlanmış sistemle benzer bir davranış gösterdiği görülmüştür. Şekil 10 ve 11'de verilen grafiklerden görüleceği üzere 24 ve 20 bit uzunluğuna sahip kayan noktalı sayılar kullanılarak tasarlanan sistemin Şekil 4 ile verilen sonuçlarla oldukça benzer olduğu görülmüştür.

16 bit uzunluğuna sahip kayan noktalı sayılarla tasarlanan sistemin de benzer bir davranış göstermiştir. Fakat Şekil 12'de verilen grafikten anlaşılacağı üzere özellikle tepe noktalarda bozulmalar göstermektedir.

12 bit uzunluğuna sahip kayan noktalı sayılarla tasarlanan sistemin sonuçlarının oldukça kötü olduğu görülmüştür. 5 bit uzunluğunda olan kesir kısmının ifade aralığının dar olması bu durumun ana sebebidir. Şekil 8 ve Şekil 9 ile verilen grafiklerden görüleceği üzere kesir kısmının dar olması, işaret gösteriminde hatalara neden olmaktadır.

Donanım tarafına bakıldığında referans alınan IEEE 754 standardında 32 bit olarak sentezlenen sistemin beklendiği üzere kaynak tüketimi açısından ilk sırada yer aldığı görülmüştür.

Tablo 2' de verilen sonuçlar incelendiğinde en az kaynak tüketiminin bit sayısının azaltılmasından dolayı 12 bit olarak tasarlanan sisteme ait olduğu anlaşılmıştır.

Bit uzunluğu azaltılarak gerçekleştirilen tasarımlardan 24 bit uzunluğundaki sistemin 32 bit sisteme göre yaklaşık %30 daha az kaynak tükettiği, 20 bit olarak tasarlanan sistemin ise yaklaşık %50 daha az kaynak tükettiği görülmüştür.

Şekil 11 ve Tablo 2 göz önüne alındığında; kaynak tüketimi ve hesaplama sonuçlarına göre 20 bit olarak tasarlanan sistemin en iyi tasarım olduğu görülmüştür.

Çalışmada ayrık zamanlı kayan kipli kontrol yaklaşımı FPGA üzerinde kodlanarak koşturulmuş ve donanım üzerinde kontrol açısından hangi nümerik gösterimin daha avantajlı olabileceğine dair çıkarsamalarda bulunulmuştur. Bu çalışmanın devamında FPGA üzerinde görsel servolama uygulaması ele alınacaktır.

5. Teşekkür

Bu bildiri TÜBİTAK-ARRS (Slovenya) 2508 programında 114E954 referans numaralı proje kapsamında desteklenmiştir.

Yazarlar Hacettepe Üniversitesi, Bilgisayar Mühendisliği Bölümü, Otonom Sistemler Laboratuvarına teşekkür eder.

Kaynakça

[1] A. Hace ve M. Franc, "FPGA Implementation of Sliding-Mode-Control Algorithm for Scaled Bilateral Teleoperation", *IEEE Trans. Ind. Inform.*, c. 9, sayı 3, ss. 1291–1300, Ağu. 2013.

[2] S. Lentijo, S. Pytel, A. Monti, J. Hudgins, E. Santi, ve G. Simin, "FPGA based sliding mode control for high frequency power converters", içinde *Power Electronics Specialists Conference, 2004. PESC 04. 2004 IEEE 35th Annual*, 2004, c. 5, ss. 3588–3592 Vol.5.

[3] N. Li, X. Lin-Shi, P. Lefranc, E. Godoy, ve A. Jaafar, "FPGA based sliding mode control for high frequency SEPIC", içinde *2011 IEEE International Symposium on Industrial Electronics (ISIE)*, 2011, ss. 1575–1580.

[4] F.-J. Lin, D.-H. Wang, ve P.-K. Huang, "FPGA-based fuzzy sliding-mode control for a linear induction motor drive", *Electr. Power Appl. IEE Proc. -*, c. 152, sayı 5, ss. 1137–1148, Eyl. 2005.

[5] F.-J. Lin, C.-K. Chang, ve P.-K. Huang, "FPGA-Based Adaptive Backstepping Sliding-Mode Control for Linear Induction Motor Drive", *IEEE Trans. Power Electron.*, c. 22, sayı 4, ss. 1222–1231, Tem. 2007.

[6] F.-J. Lin, J.-C. Hwang, P.-H. Chou, ve Y.-C. Hung, "FPGA-Based Intelligent-Complementary Sliding-Mode Control for PMLSM Servo-Drive System", *IEEE Trans. Power Electron.*, c. 25, sayı 10, ss. 2573–2587, Eki. 2010.

[7] A.-M. Lienhardt, G. Gateau, ve T. A. Meynard, "Digital Sliding-Mode Observer Implementation Using FPGA", *IEEE Trans. Ind. Electron.*, c. 54, sayı 4, ss. 1865–1875, Ağu. 2007.

[8] F. Piltan, I. Nazari, S. Siamak, ve P. Ferdosali, "Methodology of FPGA-Based Mathematical Error-Based Tuning Sliding Mode Controller", *Int. J. Control Autom.*, c. 5, sayı 1, ss. 89–117, Mar. 2012.

[9] Y. Lee, Y. Choi, S.-B. Ko, ve M. H. Lee, "Performance analysis of bit-width reduced floating-point arithmetic units in FPGAs: a case study of neural network-based face detector", *EURASIP J Embed. Syst*, c. 2009, ss. 4:1–4:11, Oca. 2009.